小马哥的 Java 项目实战营

Java EE 项目 - 第三节 数据存储之 JPA

小马哥（mercyblitz）

我是谁？

小马哥（mercyblitz）

- 父亲
- Java 劝退师
- Apache Dubbo PMC
- Spring Cloud Alibaba 架构师
- 《Spring Boot 编程思想》作者

Java Persistence API（JPA）

- 简介

JPA 1.0 整合查询语言（Query）和对象关系映射（ORM）元数据定义。

JPA 2.0 在 1.0 的基础上，增加 Criteria 查询、元数据API以及校验支持。

2009 年 JPA 2.0 最终规范发布（JSR-317）

2006 年 JPA 1.0 最终规范发布（JSR-220）

EJB 3.0 的子规范

Java Persistence API（JPA）

- 实体（Entities）

轻量级持久化域（Domain）对象。实体类可能利用辅助类或者用于表示状态。

- 约束
  - 实体类必须使用@Entity标注或者XML描述
  - 实体类至少包含一个默认构造器，并且构造器必须是public 或者 protected
  - 实体类必须是顶级类，不能是枚举或者接口
  - 实体类禁止是final类
  - 实体支持继承、多态关联以及多态查询

Java Persistence API（JPA）

- 实体持久字段和属性

实体持久状态由字段（Fields）或者属性（Properties），字段即实例的属性或变量，属性则是JavaBeans实例的setter或getter方法。

实例属性的访问性必须是private、protected或者包可见，属性的可见性必须是public或者protected。

字段和属性可能是单一类型值或集合类型值。

Java Persistence API（JPA）

- 持久字段和属性类型
  - 原生类型
  - Java Serializable类型
  - 自定义类型（实现Serializable接口）
  - 枚举
  - 实体类型（包括集合实体类型）
  - 嵌入类型

Java Persistence API（JPA）

- 字段和属性访问类型（Access Type）
  - 默认访问类型
    - 非 transient 或者@Transient字段
    - 非@Transient 属性
  - 显示访问类型
    - 注解类型
      - 实体类
      - 映射超类
      - 嵌套类
    - 注解
      - @Access(AccessType.FIELD) 字段
      - @Access(AccessType.PROPERTY) 属性

Java Persistence API（JPA）

- 实体主键（Primary Key）默认访问类型
  - 每个实体必须存在主键，主键必须定义在实体类。
    - 简单主键
      - @Id
    - 复合主键
      - @EmbeddedId
      - @IdClass

Java Persistence API（JPA）

- 实体关系

实体关系可能一对一、一对多、多对一或多对多，这些关系是多态性的，可以是单向或者双向。

- 注解表述方式
  - @OneToOne
  - @OneToMany
  - @ManyToOne
  - @ManyToMany

Java Persistence API（JPA）

- 实体双向关系

  实体双向关系是指两实体之间不仅存在拥有方（owning），也存在倒转方（inverse）。主方决定了更新级联关系到数据库。

- 规则
  - 倒转必须通过@OneToOne、@OneToMany或者@ManyToMany中的mappedBy属性方法关联到拥有方的字段或者属性。
  - 一对多、多对一双向关系中的多方必须是主方，因此@ManyToOne 注解不能指定mappedBy属性方法。
  - 双向一对一关系中，主方相当于包含外键的一方。
  - 双向多对多关系中，任何一方可能是拥有方。

Java Persistence API（JPA）

- 实体双向关系：一对一（OneToOne）

假设：

- 实体A引用单个实体B的实例
- 实体B引用单个实体A的实例
- 实体A在关系中处于拥有方

默认映射

- 实体A被映射到数据表A
- 实体B被映射到数据表B
- 表A包含一个外键关联B

举例：客户（Customer）与信用卡（Credit Card）的关系

Java Persistence API（JPA）

- 实体双向关系：多对一（ManyToOne）/一对多（OneToMany）

假设：

- 实体A引用单个实体B的实例
- 实体B引用多个实体A的实例
- 实体A在关系中处于拥有方

默认映射

- 实体A被映射到数据表A
- 实体B被映射到数据表B
- 表A包含一个外键关联B

举例：店铺（Store）与客户（Customer）的关系

Java Persistence API（JPA）

- 实体双向关系：多对多（ManyToMany）

假设：

- 实体A引用多个实体B的实例
- 实体B引用多个实体A的实例
- 实体A在关系中处于拥有方

默认映射

- 实体A被映射到数据表A
- 实体B被映射到数据表B
- 存在一个名为A_B的关联表（拥有方表名为前缀），其中包含两个外键列，一列关联表A的主键，另外一列关联表B的主键

Java Persistence API（JPA）

- 实体单向关系：一对一（OneToOne）

假设：

- 实体A引用单个实体B的实例
- 实体B没有引用单个实体A的实例
- 实体A在关系中处于拥有方

默认映射

- 实体A被映射到数据表A
- 实体B被映射到数据表B
- 表A包含一个外键关联B

Java Persistence API（JPA）

- 实体单向关系：一对多（OneToMany）

假设：

- 实体A引用多个实体B的实例
- 实体B没有引用单个实体A的实例
- 实体A在关系中处于拥有方

默认映射

- 实体A被映射到数据表A
- 实体B被映射到数据表B
- 存在一个名为A_B的关联表（拥有方表名为前缀），其中包含两个外键列，一列关联表A的主键，另外一列关联表B的主键

Java Persistence API（JPA）

- 实体单向关系：多对多（ManyToMany）

假设：

- 实体A引用多个实体B的实例
- 实体B没有引用单个实体A的实例
- 实体A在关系中处于拥有方

默认映射

- 实体A被映射到数据表A
- 实体B被映射到数据表B
- 存在一个名为A_B的关联表（拥有方表名为前缀），其中包含两个外键列，一列关联表A的主键，另外一列关联表B的主键

Java Persistence API（JPA）

- 实体继承（Inheritance）

实体可继承其他实体。实体之间支持继承、多态关联、多态查询

继承方式

- 继承抽象实体类
  - @Inheritance
- 继承已映射父类型
  - @MappedSuperclass
  - @AssociationOverride
- 继承非实体类型

Java Persistence API（JPA）

- 实体操作（Operations）

实体管理器 - EntityManager

```
@Service
public class CustomerService {

    @PersistenceContext
    private EntityManager entityManager;

    public void addCustomer(Customer customer) {
        entityManager.persist(customer);
    }

}
```

Java Persistence API（JPA）

- 实体实例生命周期（Life Cycle）
- 创建
- 持久化
- 移除
- 同步到数据库
- 刷新实例
- 淘汰

Java Persistence API（JPA）

- 持久化上下文使用期限（Persistence Context Lifetime）
- 类型
  - 事务类型（默认）
  - 扩展类型
- 阶段
  - 事务提交阶段
    - 事务类型：实体状态->脱管
    - 扩展类型：实体状态->继续维持
  - 事务回滚阶段
    - 实体状态->脱管

Java Persistence API（JPA）

- 实体监听器和回调方法
- 实体监听器 - @EntityListeners
- 回调方法
  - @PrePersist
  - @PostPersist
  - @PreRemove
  - @PostRemove
  - @PreUpdate
  - @PostUpdate
  - @PostLoad

Java Persistence API（JPA）

- 实体监听器和回调方法
- 实体监听器 - @EntityListeners
- 回调方法
  - @PrePersist
  - @PostPersist
  - @PreRemove
  - @PostRemove
  - @PreUpdate
  - @PostUpdate
  - @PostLoad

THANKS!